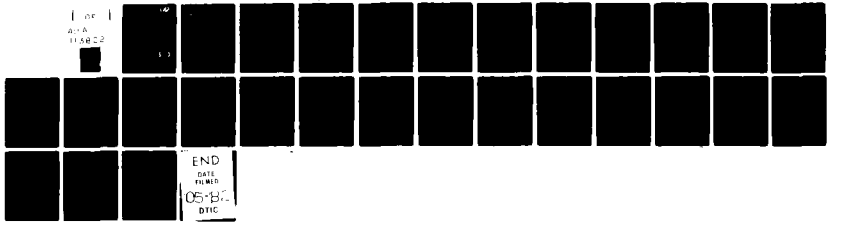
AD-A113 822 MASSACHUSETTS INST OF TECH CAMBRIDGE RESEARCH LAB OF--ETC F/G 9/3
CONVERSION OF ALGORITHMS TO CUSTOM INTEGRATED CIRCUIT DEVICES.(U)
MAR 82 J ALLEN F49620-80-C-0073

UNCLASSIFIED AFOSR-TR-82-0309 NL



END
DATE FILMED
05-82
DTIC

FINAL REPORT

Conversion of Algorithms to Custom Integrated Circuit Devices

AFOSR Contract F49620-80-C-0073

covering the period

15 April 1980 - 14 March 1981

submitted by

Jonathan Allen

March 8, 1982

Massachusetts Institute of Technology
Research Laboratory of Electronics
Cambridge, MA 02139

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFOSR-TR- 82-0309	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) CONVERSION OF ALGORITHMS TO CUSTOM INTEGRATED CIRCUIT DEVICES		5. TYPE OF REPORT & PERIOD COVERED Final Report 4/15/80-3/14/81
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Jonathan Allen		8. CONTRACT OR GRANT NUMBER(s) F49620-80-C-0073
9. PERFORMING ORGANIZATION NAME AND ADDRESS Research Laboratory of Electronics Massachusetts Institute of Technology Cambridge, Massachusetts 02139		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61102F 2305/B1
11. CONTROLLING OFFICE NAME AND ADDRESS AFOSR/NE Bolling AFB, Washington, DC 20332		12. REPORT DATE Mar 1982 13. NUMBER OF PAGES 29
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution limited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This project studies the way in which high level functional descriptions can be converted through a succession of representations by means of local transformations to a final mask specification. A major focus of the work is the determination of the form of these representations, as well as the nature of the transformations between them. Major emphasis has been devoted to the specification of high level algorithms, in a way such that performance variations can be studied, either through the use of constraint representations,

(continued)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

or by use of conflict avoidance schemes that permit the systematic exploration of various degrees of parallelism inherent in an algorithm. Following architectural determination through exploration of space/time tradeoffs, various well-formedness tests are executed, including topology extraction, circuit extraction, logic verification, design rule checking, timing analysis, and circuit simulation. In addition, testability is characterized and test vectors are generated. Following the well-formedness checks, modules can then be juxtaposed by means of placement and routing algorithms, which are under development. Emphasis is also placed on techniques for generation of modules, either through text layout languages, interactive graphics, or the construction of programs as types which are able to generate parameterizable layout.

Using these techniques it is possible to generate modules, ascertain their well-formedness, connect them into clusters at the next higher level of the overall system hierarchy, and continue iteratively until the entire design is complete. The various tasks in this contract are devoted to specifying the nature of all the techniques in this process. Several large chips have been designed using these techniques, so as to ascertain the correctness and usefulness of the design techniques we have developed.

TABLE OF CONTENTS

I. OVERVIEW

In order to place the research done under this contract in perspective, it is useful to describe the view of integrated circuit design that motivates this work. We see the design of custom integrated circuits as a set of transformations between a corresponding interleaved set of representations, starting at a high level functional description and passing through architectural, logical, circuit, device and layout representations, so that the final mask specification can be interpreted by the fabrication process (in our case single-level metal silicon gate NMOS) to produce chips that will satisfy the originally intended functionality. Important issues are the nature of the computational representation for each of these levels and the algorithmic transformations between levels that preserve the representation of functional intent that is manifest at each level. For example, conversion of an architectural (block diagram) representation to a more primitive set of logical gates, together with their interconnection network, must preserve all of the original architectural goals while implementing them in ways that are appropriate to gate level logic. We call the means to ascertain these transformational constraints a set of well-formedness checks and a major thrust of our effort is the design of programs to look for these correctness constraints.

In addition to the transformational view between representations that we have described above, however, we need to be able to deal with the size complexity of a design through techniques of regularity, modularity and hierarchy. Regularity is an important notion because it permits the use of canonical design forms, such as PLAs and register arrays, that cut down the

amount of unique cells that have to be designed by means of iterative techniques. Modularity refers to the characteristic of an overall chip design, whereby it can be regarded as a set of individually well-formed modules which are then interconnected by means of the specified hierarchy in a way that preserves this well-formedness, while possibly modifying the boundaries of the modules in order to accept the constraints of the hierarchical interconnect. It is the responsibility of the architectural specification to break down the overall system specification into this set of hierarchically related modular elements.

The view that emerges from this perspective is summarized in figure 1, which we have found helpful as a framework to relate the various aspects of our work. We conceive that the high level functional description of the system is given in terms of some hardware design language, possibly augmented by a microcode specification to establish the semantic use of the hardware facilities. At this level, it is appropriate to examine space/time tradeoffs, whereby greater speed can usually be attained at the expense of additional hardware (and hence, parallelism). Once the architectural design is fixed, it specifies a set of modules (memories, arithmetic logic units, register files, program logic arrays, etc.) that have to be designed. It is important to realize that the time at which these elements are designed can vary widely, as suggested by the figure.

Of course, some cells may have been completely designed previously, as is typical of the standard cell approach. In most design styles, there will always be some cells, such as pads, that are given (i.e. previously designed and checked for correctness). These elements can clearly be used directly, it

being only necessary to retrieve the design specification from some appropriate library. In other cases, the desired cell is of a type (e.g., program logic array, shift register, memory), for which a program is available to generate the design. We call such cells parameterized to indicate that they cannot be retrieved directly from a library, but they can be produced by a program (previously constructed) that can generate the cell specification from a set of parameters that are generally related to the functional specification (e.g. logic) of the cell. PLA generator programs are examples of such very useful techniques and they clearly provide for the fast and correct design of many different cells, albeit of a restricted set of types. These types can generally be regarded as canonical forms that make substantial use of repeated cells, which are sometimes slightly modified as dictated by the parameters. The design of cells by writing programs to design them is a powerful idea, and one that we are increasingly exploiting. Finally, if we cannot find an appropriate cell design in the library, or a program design to produce the desired type of cell, then the cell must be laid out from scratch. For this purpose, layout programs (either textually or graphically based) are required and we have experimented with a number of these interactive programs, each providing its own design style emphasis.

Given a set of cells designed by the techniques described above, they must be checked for well-formedness, as indicated in the center of figure 1. The checks are keyed to the various levels of representation that we have described above. Thus, microcode simulation is used at the functional level, and design rule checks are used at the artwork level. From the artwork, it is possible to transformationally derive both a topological representation, useful

for both connectivity checks and switch-level logic simulation, and an equivalent circuit model, useful for timing verification. The circuit model, of course, requires accurate transistor device models as part of its representation, but derives the characterization of the interconnect network directly from the artwork specification. Given the circuit model, it is possible to perform circuit simulation and different degrees of precision can be provided by utilization of varying approximations for both the device models and the circuit parameters. An important aspect of well-formedness is suitability for testing. This is a very unsettled area at present, in part due to poor understanding of MOS failure modes, in part due to reluctance to provide on chip capability for testing, and in part due to the increasing complexity of IC designs. So far, promising techniques have centered around the ability to set and read the entire state of a chip, the use of modulo arithmetic and signature analysis, and the ability to test a module from its terminals without knowledge of its internal gate implementation, which, of course, is not strictly implied by the functional constraints of the module.

Once we know that all the modules are well-formed, it remains to place them with respect to each other and establish the requisite interconnection paths. That is, we must orient a set of orthogonally related rectangles (modular elements) and route their interconnect so as to occupy minimum area in a given technology. We emphasize that this is a combinatoric problem of great complexity, and heavily dependent on the technology employed. Some design styles, such as the standard cell technique, are sufficiently constrained to permit complete routing algorithmically, but the general case of orthogonally related rectangles of arbitrary size is exceedingly difficult and is still under study by us and others.

MODULAR, HIERARCHICAL DESIGN

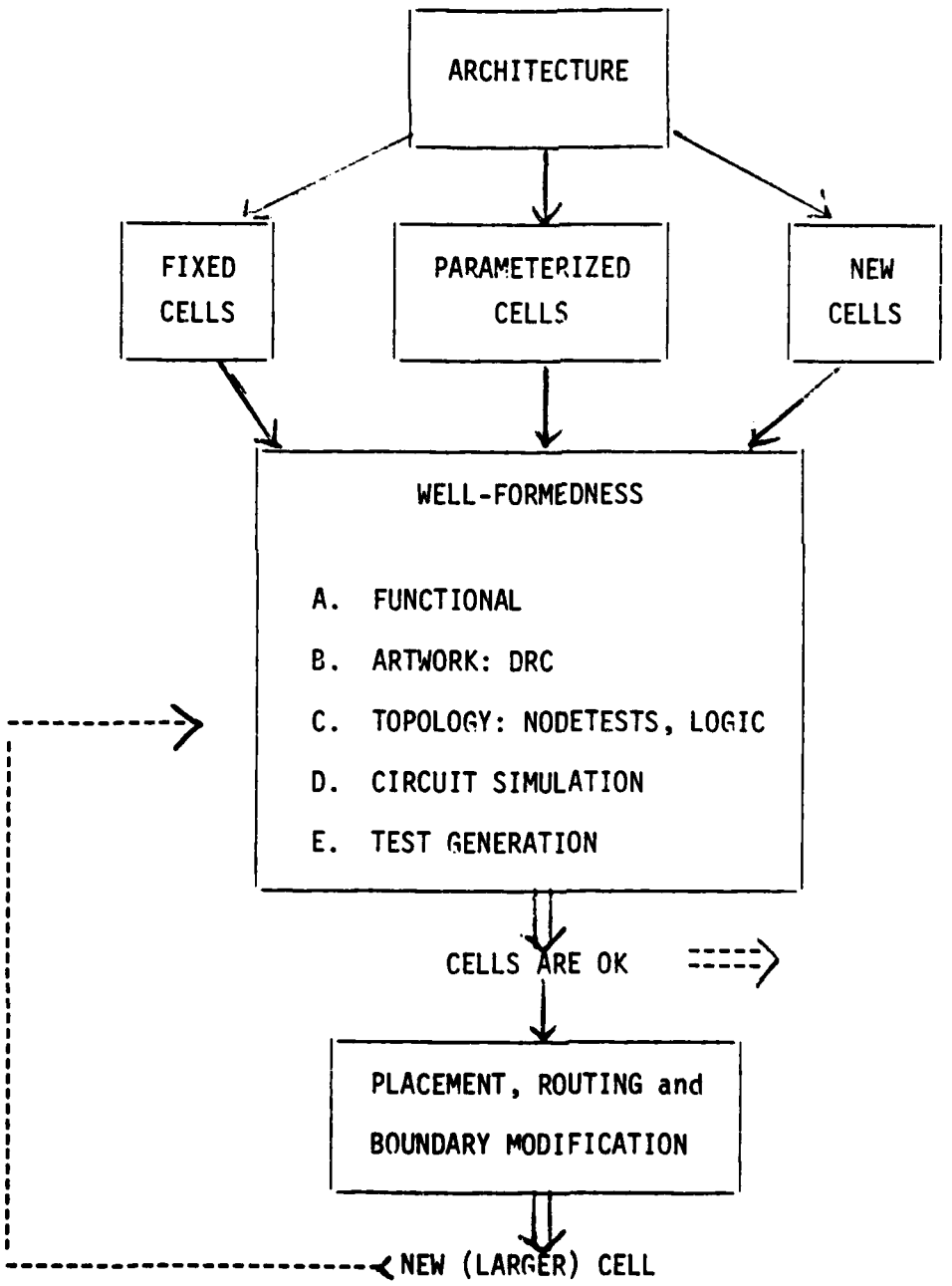


FIGURE 1

Interconnections will naturally be grouped in accordance with the hierarchy of the architecture, so as the modules are clustered into larger units, it is necessary to test the well-formedness of each such cluster. Rather than repeat previous tests, it is usually sufficient to make these checks only at the periphery of the constituent modules in a cluster. This gives rise to the notion of hierarchical design rule checking, which we have developed conceptually, and are implementing in both software and hardware.

Following figure 1, we can see that as the modules of a system are grouped and connected, we can follow up the hierarchical organization of the architecture until the entire design is assembled and tested for well-formedness. At this point, the design is complete and we are done.

In the succeeding sections of this report, we state the projects undertaken during this contract and report on our progress in the context of this overview. In this way, we hope to clarify the intention of our various activities, and show how they lead to a uniform design system methodology.

II. STATEMENT OF WORK

Following the proposal for this contract, we describe here the projects that comprise the work undertaken. First, a major effort has been continued to develop the representation of constraints for two different purposes. One of our fundamental aims is to separate an algorithm into its competence (which specifies what the algorithm does) and its performance (which specifies how the algorithm is computed). Unfortunately, the current literature confounds these two attributes of an algorithm in such a way that it is difficult to manipulate the performance, while keeping the competence invariant. VLSI presents the custom circuit designer with many opportunities for exploring performance variations, but this designer needs mechanisms to make sure that while the performance options are explored, the competence of a design remains invariant. We see the characterization of these two attributes of algorithms to be a fundamental problem in computer science, as well as being of direct utility to the design of custom integrated circuits. Our idea has been to attempt to represent the competence of broad classes of systems by constraint networks, which should capture the competence of these systems. Performance strategies could then be erected on top of these constraint networks in various ways so that while the competence is never perturbed, a variety of performances may be obtained. We have been successful in representing electrical circuits by means of constraint networks, and these networks can be generalized to any system that can be characterized by a set of simultaneous linear algebraic equations. Although it is by no means clear how such representational techniques can be extended to other classes of systems we continue to look for this possibility, realizing that this problem will not yield easily. Perhaps a more obvious use

of constraints is found in the maintenance of design rules over a layout, and somewhat more abstractly in the maintenance of proper timing disciplines at the system architecture level. One of our graphic editing systems for layout provides built-in constraint maintenance so that design rule violations can be interactively provided to the designer, as the layout is developed. Another example of the use of constraints is in the specification of transistor sizes where specified W/L ratios must be maintained. For example, if either W or L is specified, then the other is constrained because of the specified ratio. It is possible to maintain this constraint automatically in the layout system and we have been exploring the use of constraints for this purpose. In order to use constraints in a flexible way, it is important to be able to represent them formally in such a way that they can naturally be cascaded to form broader constraint systems. This is a major programming project and has occupied much of our time during the contract period.

Instead of separating algorithms into the two attributes of competence and performance, an alternative procedure is to start with a particular specification of a design, as specified in a high level functional representation, and then provide formal means for changing this specification in a way that cannot disturb the underlying competence, even though this competence is never identified explicitly. The key to this procedure is to recognize within such formal specifications the presence of sequential conflict which prohibits the use of parallelism. We have shown that it is formally possible to scan the top level specification of a design and determine precisely where parallelism can be employed without mutilating the underlying semantics of the design. In this way, latent parallelism can be discovered and

utilized to increase architectural performance in any task. These techniques are particularly useful for the optimization of high performance signal processing tasks.

An important area of our work centers around the specification of detailed artwork. For this purpose, we have devised a number of different layout languages which have a variety of different features. In one approach, the advantages of high level specification using APL has been exploited to provide a concise and easy to use layout language. This language is also useful for serving as the base upon which other programs can be built that utilize the layout language for some specialized purpose, such as the generation of program logic arrays. In another approach, we have chosen to represent layout information in terms of a prototype specified by a procedure written in the programming language LISP. In this way, a particular layout is generated when the program is run, utilizing a particular set of parameters which individualize the layout. The underlying data structure used for the LISP based programming language has been designed in such a way that it can serve as a common bridge between both text based layout and graphic based layout. We feel strongly that it is important that any layout information entered into a system graphically be obtainable in textual form, and vice-versa. We are also investigating the use of interactive graphics for small cell design based on a menu driven approach. Many of the problems in this area pertain to human factors, and we are steadily modifying this system to reflect our experience.

As indicated in the overview above, one of our goals is to provide a broad variety of well-formedness tests so as to insure that the design supplied

to the mask maker is correct at all representational levels. One of the most important checks is for design rule correctness, and we have explored three different methods for implementing design rules. The most straightforward technique is to utilize rectangle descriptions of the layout, and to base all tests on this formulation. More recently, however, we have developed design rule checking tests based on a course grid layout, which has many advantages, once the overall layout is rasterized. Both of these techniques, however, can be very time consuming, and so a major part of our work has been the investigation of means to speed up these algorithms. We have devised ways to modularize the design rule checks, hence reducing computation time from a squared to a linear dependence on the number of rectangles involved in the design. Perhaps our most interesting project for reducing design rule checking time is the design and construction of special purpose hardware for design rule checking. This project is well underway and promises to reduce the time for this activity by at least two orders of magnitude.

It is exceedingly important to verify that an integrated circuit performs the desired logic given by the high level specification. For this purpose, we have undertaken an extensive project for the construction of a switch level simulator that is appropriate to the MOS technology. We have shown that conventional logic simulators that were developed for TTL use are very awkward, if not incorrect, when used for MOS circuits, and so we have taken a basic approach to the design of MOS logic simulators. This work has led to a highly useful simulator, but also to an entirely new theory of MOS digital systems, which serves as the basis for this simulator. In this way, we have contributed not only a practically useful simulator, but also the

conceptual underpinnings that are essential for a proper logical perspective on MOS digital systems. The input to the logic simulator is a topological representation of the circuit which can be easily extracted from the coarse grid representation of the layout. We have found that this representation itself can be used for many well-formedness checks, including connectivity tests, power supply integrity, and the maintenance of appropriate dc levels.

Once each module of the overall integrated circuit has been checked for well-formedness, it is necessary to both place and connect these modules according to the architectural hierarchy provided by the initial design. If we assume that each module is bounded by a rectangular box, and that these boxes are orthogonally related on the overall layout, then an important and difficult problem is to place these rectangles and form the interconnect between them within a minimal area. While this problem has been recognized by many researchers for several years, it has not yet been solved, and a major part of our effort is devoted towards the combinatorial techniques needed to achieve both optimal placement and routing. An initial program designed to provide the routing capability needed, once an appropriate layout of the rectangles is given, was implemented during the summer of 1981, and experience from this usage has provided the motivation for continuing improvements. We emphasize that this is an exceedingly fundamental and difficult problem and that we are approaching the overall goal by means of a sequence of programs of lesser capability, thus expecting to gain from our experience on more restricted problems in a way that will lead to the best design for the overall placement and routing tasks.

It is exceedingly useful to be able to generate basic canonical structures, such as register files, program logic arrays, and arithmetic logic units, by means of parameterizable procedures. We have developed a very successful design program for program logic arrays which accepts the logic specification as a sum of products as input, and produces a complete detailed layout as output. An exceedingly useful feature of this program is its capability to provide many useful answers to the designer prior to generating the complete layout. For example, the size of the bounding box, the delay through the PLA and its power consumption can all be provided by the program very quickly without the necessity to generate the entire layout. This program is used very heavily, and serves as a prototype for the design of other similar procedures for other canonical forms.

The various research projects mentioned above within this section have all led to implemented programs on one of two facilities which we are constructing. The most general facility is based on a DEC system 20 with Hewlett Packard graphic terminals and plotters provided for the designers. This facility provides an excellent software base in as many as six different languages for the various research projects, and we have written an overall executive program that knits together all of the research programs within this contract. A more specialized facility is the MIT Artificial Intelligence Laboratory LISP machine which provides excellent interactive graphic capability on a single user basis. The program Daedalus has been implemented on this facility, as well as the placement and interconnect program.

Lastly, we should mention that we believe that work on custom integrated circuit design tools should not take place in a vacuum and that it

is important that the designers of these tools, also be designers of integrated circuits. Accordingly, a substantial part of our research emphasis has been the design of large chips. These have included a special purpose microprocessor for executing a dialect of the programming language LISP, an encription chip, and a signal processing chip which is intended for speech processing. Many smaller designs have also been executed, but all of these efforts have been important in determining the direction of our overall research.

From the summary above of the work undertaken for this contract, we hope it is clear that we have addressed much of the basic structure of our conceptualized design process. The evolution of these tools is expected to lead naturally, as a product of our research, to an integrated comprehensive design system for the conversion of algorithms to layout specifications.

III. STATUS OF RESEARCH

The point of view taken by our research under this contract is well expressed in a review article, written by Profs. Allen and Penfield, titled "VLSI Design Automation Activities at MIT." This article provides both a presentation of the basic conceptual point of view underlying our research as well as a description of the various research projects going on at MIT in the design automation area, almost all of which are supported under this contract.

As described in the previous section, the formal representation of constraints so as to enable their use in programming has been a major theme of our work. Professor Sussman and his student Guy Steele, who is now on the faculty at Carnegie Mellon University, have led this effort, building upon use of these techniques in the analysis of electric circuits. Their prior work has shown that constraints can be used to propagate via causal rules results through an electrical network, both maintaining the properties dictated by these constraints and enabling the computation of system variables that depend, however indirectly, on these constraints. Sussman and Steele have generalized their techniques of constraint representation to the solution of simultaneous linear equations. Each of these equations can be thought of as representing a constraint (i.e. equality as expressed in the equation) and hence the system of constraints expresses just what has to hold and no more in the system of equations. They have shown that it is possible to apply to these constraints a number of different strategies to solve the equations, including changing the set of unknown variables. We may think of this work as showing, for a limited class of systems, a way to exhibit all possible solutions made possible by the constraints. In this way the set of all control strategies for solving the

equations is obtainable from the underlying constraint network, which remains invariant under all of these control strategies. Sussman and Steele have described these techniques in a fundamental article published in the AI Journal, referenced below. It is important to emphasize that while the results obtained up to this point are suggestive, it is not clear how general they may be, and there is still a great deal of research to be done before an adequate representation of algorithmic competence can be obtained in terms of constraints. Nevertheless, we believe this work to be fundamental and important, certainly to VLSI design and also to the theoretical basis of computer science in general. Guy Steele's doctoral thesis is devoted entirely to the formal means of constraint specification, together with many examples of constraint use. In this way, the programmatic basis for the representation of constraints is firmly established in a major research document.

Building upon the electrical circuit analogy, it is clear that constraints can be used to provide "dependency directed backtracking" in computer systems, so that it is possible in a computational system to examine the reason why any particular value or state was reached. A particular example of this technique is the tracing of the causes for design rule errors. Such an approach has been taken by Dr. Howard Shrobe in the system called Daedalus which provides for an interactive graphic editor together with the basic constraint propagation techniques developed earlier by Sussman and Steele. Daedalus is implemented on an MIT AI Lab LISP machine, using both black and white and color graphics, and provides a high quality personal VLSI design station, using the programming language LISP. The Daedalus system also uses the layout language DPL, which represents layout in terms of parameterized

procedures, that can then be run to instantiate the modular elements of a design. Another important feature of the Daedalus system is that layout information can be entered, either textually (via DPL) or graphically, and output is available in either of these two formats. Based on this experience, we feel strongly that this duality of representation must be maintained in any VLSI design system. The Daedalus system is being used to design a LISP based microprocessor chip, called SCHEME. This is an aggressive effort to compile from a high level specification a special purpose microprocessor which exploits the many attractive features of the LISP programming language in an efficient way.

We have mentioned before that while we do not yet have the means to represent algorithmic competence for all systems in terms of constraints, we do know how to manipulate a particular performance variant of an algorithm into other versions that provide different performance, but the same competence, by means of a formal analysis of conflict within a high level specification of the desired system. These formal techniques were provided in a doctoral thesis by Glen Miranker, and make it clear that given any initial architectural specification of a design, it is always possible to manipulate that design to reveal all possible performance variations, and hence all degrees of latent parallelism that may be present in the initial specification. Thus, we have the theoretical basis for architectural manipulation under the condition that the semantics (i.e. results) of the computations provided by the system do not change as a result of these manipulations. While we continue to be guided by these observations in the design of several of our systems, and in the specification of new VLSI design tools, this ability to explore architectural

variants of a design has not yet been incorporated into a custom VLSI design system, although there has been much work on the plans for such a system.

We have mentioned that one of our main interests is in the improvement of the efficiency in performing well-formedness checks at the artwork level. Modern custom integrated circuit designs may involve as many as a million transistors on one chip, and hence the number of rectangles needed to specify this level of complexity may be of the order of 10^7 or more rectangles. It is not uncommon for many integrated circuit design centers to experience great cost and delay in verifying designs at the artwork level due to this large number of artifacts. While there have been several approaches to breaking up designs, either in terms of the natural hierarchy or in terms of some forced grid hierarchy, so as to reduce the complexity of design rule calculations, the major focus of our effort has been on the design and construction of special purpose hardware for design rule checking. The doctoral thesis work of Larry Seiler has been directed towards this goal, and he has provided a complete design for such a system, that would be provided as a modular unit to a basic interactive design station. Three custom chips are needed for this design, and are being produced as part of the work of this contract. Experience with this design has shown that all artwork-based techniques can be sped up by at least two orders of magnitude by use of special purpose hardware. We are very encouraged by the status of this work and believe that it is progressing well.

A major contribution to VLSI design workers has been our work on automatic generation of program logic arrays from a high level logical specification. There are many interesting aspects to this work. Firstly, the program for generating PLA artwork is built on top of an existing layout

language, AIDS, which in turn is embedded in APL. This technique shows that it is possible to build very sophisticated systems by layering one language on top of another. Each language can make calls on the resources provided by the language below it in the hierarchy, and in this way a great deal of programming effort can be saved. Another advantage of the PLA generator is that the designer is able to obtain a great deal of useful information from the program without actually instantiating a layout. Thus, the size of the bounding box, the number and placement of location points, the power dissipation, the delay through the PLA, and several other useful parameters are quickly obtainable by the designer once the input logic specification has been furnished. We are not currently focused on problems of logic minimization, but a separate study performed by Lorne Cooper has shown an improvement over the logic minimization techniques generally described in the literature. As far as folding techniques are concerned, while we recognize that the appropriate grouping of inputs and outputs can often lead to space savings, we are not trying to mechanize these changes in terms of a program, but instead rely on the designer to recognize these opportunities. The PLA generator program, written by Professor Glasser, based on the layout language of Professor Penfield, has been widely used and described, and its features have been employed in several other designs using different programming languages.

As the dimensions of VLSI circuits are scaled down, the properties of the interconnect wires between active elements become more and more important in terms of determining the overall speed of the system. Indeed, it is sometimes remarked that the delay through interconnect may dominate the delay through active devices, so that the characterization of interconnect delay in

terms of the resistance and capacitance of these lines becomes exceedingly important to the viability of large scale circuits. While it is possible to obtain these values of resistance and capacitance by circuit extraction and then to use these values to perform circuit simulation in order to estimate the delay, we have also been focused on the less computationally taxing process of providing bounds on such delay by theoretical means applied to the characterization of the circuit interconnect itself. Professor Penfield has provided very tight and accurate bounds on such delay through fan out trees of interconnect from any given active point of the circuit. These bounds are sufficiently tight, yet easily obtained computationally, that they can provide the designer with very useful information for the overall timing structure of his design.

Once the topology of an integrated circuit layout is extracted by means of a simple algorithm, it is possible to perform an accurate switch level logic simulation which is an essential part of the overall verification of a custom design. The logic simulator which we have designed, called MOSSIM, is a so-called unit delay simulator, in that it does not provide direct timing information, but does accurately characterize the sequence of logical values at all nodes of the simulated network. There have been substantial theoretical difficulties having to do with the propagation of the unknown, or X state, but these have been worked out satisfactorily in the version of the MOSSIM program developed by Randall Bryant as part of his doctoral thesis. While this program is very useful and has been used at many companies and universities in addition to MIT, an additional fundamental contribution has been the development in Bryant's thesis of a basic theory for MOS digital design. This theory has been

necessitated by the lack of correspondence between MOS digital systems and the prevalent TTL systems, which had been simulated by many previous systems. For example, in TTL systems memory must be provided explicitly, whereas in MOS systems memory can be provided by charge stored at any node of the system, and of course, this property is frequently exploited in practical MOS designs. Another feature of MOS systems is the use of pass transistors (or transmission gates), which provide current flow in both directions. Therefore, the usual Boolean gate model used by TTL simulators is inappropriate to MOS circuits. Building upon these and other observations, Bryant has achieved a systematic characterization of MOS digital systems, and we believe that his thesis is a fundamental and lasting contribution to the characterization of MOS digital systems from both the design and simulation point of view.

Professor Rivest is leading a substantial group of students in the development of routing techniques. The major emphasis has been on the provision of algorithms for the placement and routing of orthogonally related rectangular shaped modules of arbitrary aspect ratio and size. This project is a good example of the ability of those trained in theoretical computer science to apply combinatoric techniques to the solution of these problems. At present, a program called PI has been implemented for providing the interconnect on an experimental basis, provided the layout has already been specified. We expect within the next contract period to add algorithms for the layout together with the interconnect so as to provide a unified program for placement and routing. This is an exceedingly difficult problem, and we expect to go through several iterations in the design of this program, but we are encouraged by the initial results. Another approach to the routing problem is

to provide specialized routers, such as those needed for river routing of buses between two modules. Although we have not yet developed these programs in detail, we believe that it may be possible to simplify the design of routing programs by providing a number of specialized routers, rather than trying to bring together all kinds of routing capability in one program. In fact, it may be possible to supply this routing capability in the form of an interactive program which would serve the designer as a consultant and even provide the designer with some characterization of the degree of difficulty of a requested routing task. This work is very much in its infancy, but we regard it as an interesting alternative to the more combinatorically based theoretical approach.

The MIT AI Lab LISP machine system has been used so far as the base for both the Daedalus program and the PI placement and routing program. These programs are, of course, coded in the language LISP and take advantage of the large degree of sophisticated system software that is available on this machine, and they also exploit the large address space provided. Many of our programs, however, are implemented in languages that run on our DEC System 20 computer, which is a large time-shared facility with a large throughput capacity. We have provided this system with a number of Hewlett Packard 2647 interactive graphic terminals, which are used for both our research and teaching in the integrated circuit design area. Four color graphic plotters are also provided, as are graphic printers and a digitizer. Stipple plots of layouts may also be generated on an electrostatic printer. Recently, Gary Kopec has provided an overall coordination program, called LSIAA, which enables the user to call on the resources of a wide variety of the different programs

that we have mentioned without needing to know all of the detailed interface conventions for each of these programs individually. In this way, it is possible for the designer to progress through layout to the use of the many well-formedness checks without the need to move from the overall control of this executive program. In this way, a large amount of the input/output idiosyncrasies of programs has been hidden from the eyes of the user, and has provided a comfortable interface for our design community.

Lastly, we mention that there is a large variety of custom integrated circuits being designed at MIT in the context of this contract. All of the design tools that we are developing are being actively used for the design of these chips, and hence benefit from our direct experience. In one activity, a large chip for the implementation of a dialect of LISP, called SCHEME, has been generated. A first version of this chip provided a successful example of the use of microcode compilation techniques to automatically generate a large part of the data path and control structure needed. We are currently generating a revised design of the SCHEME chip, which will provide for increased architectural performance by means of parallelism in the data path unit. This is a good example of the possibility of improving the performance of a system considerably by means of special purpose techniques, deriving from knowledge of the particular registers needed and how they are to be utilized in the LISP based system. Another large chip has been the RSA encription chip which features a very large arithmetic-logic unit, 512 bits long. This chip is a good example of a case where a custom design is needed, since one is hardly likely to find an arithmetic logic unit, 512 bits long, in any commercial chip. We expect during the next contract period to get experience with the

performance of this chip, which will undoubtedly lead to revision of some aspects of its design to iteratively gain further performance improvements. While the two chips already mentioned provide substantial architectural performance, in another effort we are attempting to provide substantial circuit performance together with architectural performance due to the need to satisfy real time requirements. This third chip is intended to perform speech synthesis tasks, although it can also be used for a wider variety of signal processing tasks. We feel very strongly that it is important to be able to provide both architectural and circuit performance in designs, and an increasing emphasis under this contract is to develop both the means for characterizing circuit performance, as well as the capability to generate it in the context of the overall design system. It is to be expected that an increasing amount of effort under this contract, both in terms of design tools and in terms of chips designed will be focused on high performance signal processing tasks, as this provides both a good test bed for our design ideas as well as a wide variety of very useful circuits for military applications.

IV. PUBLICATIONS

In the list below, we cite several publications that describe work done during the contract period. Readers wishing detail on any activities under this contract should consult directly with Prof. Allen.

1. Allen, J. and Penfield, P., Jr., "VLSI Design Automation Activities at M.I.T.," IEEE Transactions and Systems, Vol. CAS-28, no. 7, July 1981, p. 645.

2. Allen, J., "Conversion of Algorithms to Custom Integrated Circuits: An M.I.T. Perspective," Proceedings, IEEE 1980 International Conference on Circuits and Computers, Port Chester, NY, October 1-3, 1980.

3. Glasser, L.A. and Penfield, P., Jr., "An Interactive PLA Generator as an Archetype for a New VLSI Design Methodology," Proceedings, IEEE 1980 International Conference on Circuits and Computers, Port Chester, NY, October 1-3, 1980.

4. Seiler, L., "Special Purpose Hardware for Design Rule Checking," also Proceedings, Second Caltech Conference on Very Large Scale Integration, California Institute of Technology, January 19-21, 1981.

5. Bryant, R.E., "MOSSIM: A Switch-Level Simulator for MOS LSI," Proceedings, 18th Design Automation Conference, Nashville, Tennessee, June 29-July 1, 1981.

6. Bryant, R.E., "A Switch-Level Simulation Model for Integrated Logic Circuits," Ph.D. Dissertation, Department of Electrical Engineering and Computer Science, M.I.T., June 1981.

7. Sussman, G.J. and Steele, G., Jr., "Design of LISP-based Processors," Communications of the ACM, 23 (11), 628-45, November 1980.

8. Sussman, G.J. and Steele, G., Jr., "Constraints: A Language for Expressing Almost-Hierarchical Descriptions," Artificial Intelligence 14 (1), 1-39, August 1980.

V. PROFESSIONAL PERSONNEL

Profs. J. Allen

L. Glasser

P. Penfield, Jr.

R. Rivest

G. Sussman

Dr. H. Shrobe

Mr. R. Greenblatt

VI. THESES AWARDED

Steele, G., "Definition and Implementation of a Computer Program Language Based on Constraints," M.I.T. Ph.D., July 1980.

Bryant, R.E., "A Switch-Level Simulation Model for Integrated Circuits," M.I.T. Ph.D., June 1981.

Cooper, L.J., "Logical Optimization of Programmable Logic Arrays," M.I.T. B.S., July 1981.